Design Overview for *"International Space Station Tracking"*

Name: Thanh Tam Vo
Student ID: 103487596

Summary of Program

Inspiring by the website Open Notify (open-notify.org), I created a Console Application that show the current position of the International Space Station of NASA. Also, based on the latitude and longitude of the spacecraft, the program will send HTTP request to the website GeoNames and return the name of the position (as JSON format), it could be a country name, or an ocean name.

This simple program has 3 functions:
+ Showing the current position of the International Space Station
+ Listing all the astronauts in the database, the program will show their names and their current craft.
+ Showing the current position of the International Space Station for 10 times.

Simple output of the program:

```
Type '1' to track the current position of the International Space Station
Type '2' to list all the astronaut in the database
Type '3' to track the International Space Station for 10 times
Your command:
1
The spaceship is currently at 30.1800, 141.0784 (over Philippine Sea)
```

Required Roles

Describe each of the classes, interfaces, and any enumerations you will create. Use a different table to describe each role you will have, using the following table templates.

Table 1: Object_Management.cs

Responsibility	Type Details	Notes
Managing application objects by their IDs	List<string> : _listOfObject	

Table 2: Application_Object.cs

Responsibility	Type Details	Notes
Providing common features of the application objects	name : string isHuman: bool	This is an abstract class It also inherits from the Object_Management class

Table 3: Craft.cs

Responsibility	Type Details	Notes
Demonstrating the International Space Station object in the program.	longitude : string latitude : string	It inherits from the Application_Object class

Table 4: Astronaut.cs

Responsibility	Type Details	Notes
Demonstrating the astronaut object in the program.	craft : Craft	It inherits from the Application_Object class

Table 5: Command.cs

Responsibility	Type Details	Notes
Demonstrating the user command to the application		This is an abstract class

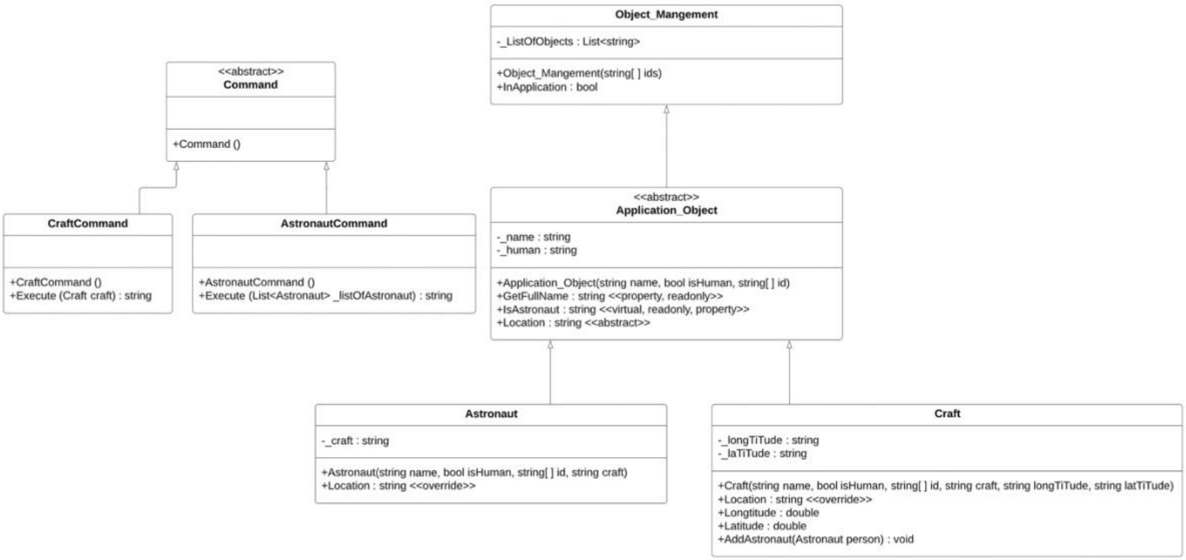
Table 6: AstronautCommand.cs

Responsibility	Type Details	Notes
Demonstrating the user command to the application for the astronaut	Function Execute : string	It inherits from the Command class There is only a function and it will return the list of all astronauts in the data set

Table 7: CraftCommand.cs

Responsibility	Type Details	Notes
Demonstrating the user command to the application for the spaceship	Function Execute : string	It inherits from the Command class There is only a function and it will return the current position of the spaceship

Expected UML Diagram


Object_Mangement
-_ListOfObjects : List<string>
+Object_Mangement(string[ ] ids)
+InApplication : bool
<<abstract>>
Command
+Command ()
CraftCommand
+CraftCommand ()
+Execute (Craft craft) : string
AstronautCommand
+AstronautCommand ()
+Execute (List<Astronaut> _listOfAstronaut) : string
<<abstract>>
Application_Object
-_name : string
-_human : string
+Application_Object(string name, bool isHuman, string[ ] id)
+GetFullName : string <<property, readonly>>
+IsAstronaut : string <<virtual, readonly, property>>
+Location : string <<abstract>>
Astronaut
-_craft : string
+Astronaut(string name, bool isHuman, string[ ] id, string craft)
+Location : string <<override>>
Craft
-_longTiTude : string
-_laTiTude : string
+Craft(string name, bool isHuman, string[ ] id, string craft, string longTiTude, string latTiTude)
+Location : string <<override>>
+Longtitude : double
+Latitude : double
+AddAstronaut(Astronaut person) : void

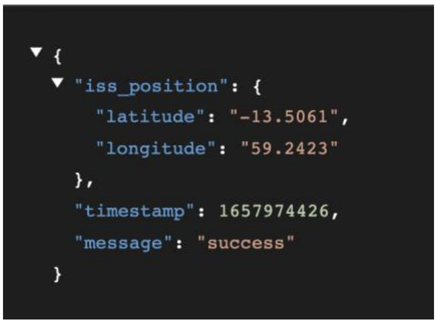
(Screenshot from LucidChart on July 16th, 2022)

Overview of program structure

1. User input command: track the position of the International Space Station or list all the astronauts in the data set.
2. Depending on the user input, the program will send the HTTP request to the website for getting data and initialize the object.
3. Program show the result (Spacecraft position or List of astronauts)

***Note:** This program uses Pure Fabrication concept by storing the JSON data in a temporary class.

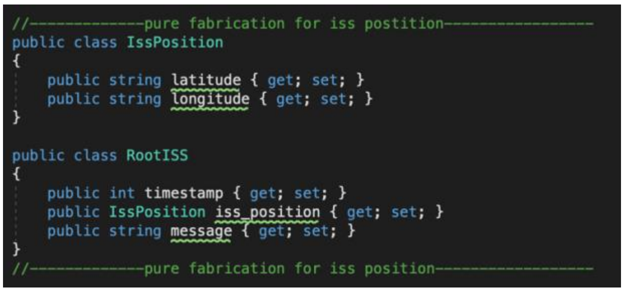
The JSON format of the International Space Station does not change, according to [Open Notify (open-notify.org)](open-notify.org), the JSON format of ISS is shown below



```
{
  "iss_position": {
    "latitude": "-13.5061",
    "longitude": "59.2423"
  },
  "timestamp": 1657974426,
  "message": "success"
}
```

JSON format of ISS
(Captured from Open Notify)

Therefore, inside the Main program, I created two temporary class to store the data above



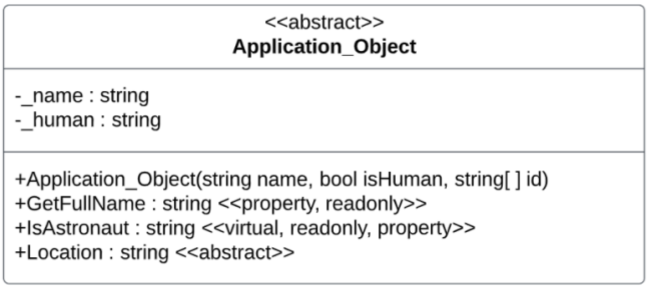
```
//------------pure fabrication for iss postition----------------
public class IssPosition
{
    public string latitude { get; set; }
    public string longitude { get; set; }
}

public class RootISS
{
    public int timestamp { get; set; }
    public IssPosition iss_position { get; set; }
    public string message { get; set; }
}
//------------pure fabrication for iss position-----------------
```

The application of pure fabrication in the program
(Captured from my Visual Studio)

Abstraction in program

The program will have an abstract class, which is the class called *Application_Object,* by indicating the common feature of the application object such as: name, is it a human or not, its location, etc.



<<abstract>>
Application_Object

-_name : string
-_human : string

+Application_Object(string name, bool isHuman, string[ ] id)
+GetFullName : string <<property, readonly>>
+IsAstronaut : string <<virtual, readonly, property>>
+Location : string <<abstract>>

Application_Object abstract class
(Captured from LucidChart)

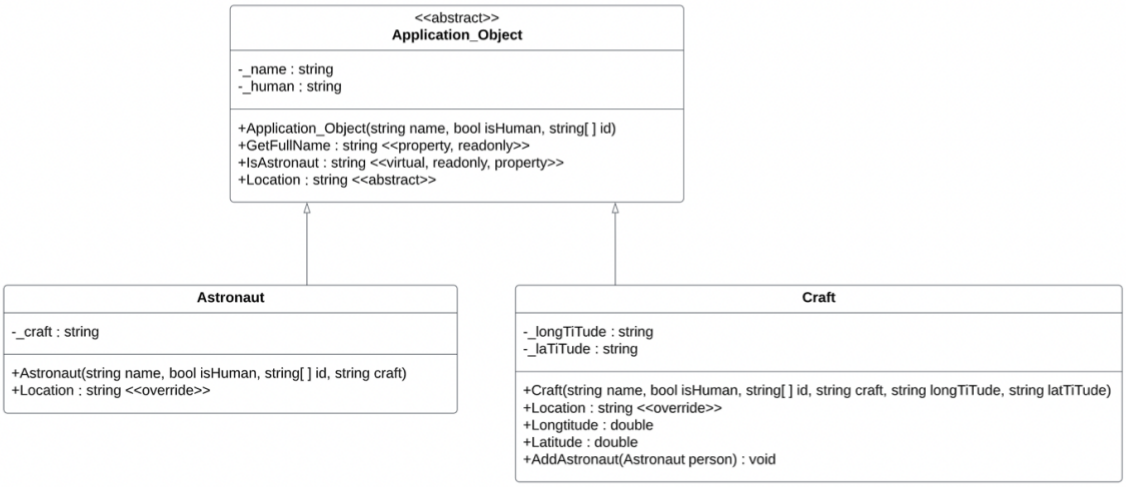
Inheritance and Polymorphism in program

There are two classes that inherit from the *Application_Object* class, they are *Craft* and *Astronaut* because both are main objects of this console application.

Both have the *Location()* function (because they inherit from the *Application_Object* class), however, each class performs differently.

+The *Location()* function of the *Craft* class will return the craft's latitude and longitude
+ The *Location()* function of the *Astronaut* class will return the astronaut's current craft

The UML diagram for those classes is depicted below


<<abstract>>
Application_Object
-_name : string
-_human : string
+Application_Object(string name, bool isHuman, string[ ] id)
+GetFullName : string <<property, readonly>>
+IsAstronaut : string <<virtual, readonly, property>>
+Location : string <<abstract>>
Astronaut
-_craft : string
+Astronaut(string name, bool isHuman, string[ ] id, string craft)
+Location : string <<override>>
Craft
-_longTiTude : string
-_laTiTude : string
+Craft(string name, bool isHuman, string[ ] id, string craft, string longTiTude, string latTiTude)
+Location : string <<override>>
+Longtitude : double
+Latitude : double
+AddAstronaut(Astronaut person) : void

UML diagram to demonstrate the use of Inheritance and Polymorphism
(Captured from LucidChart)